

Beatrice Folino

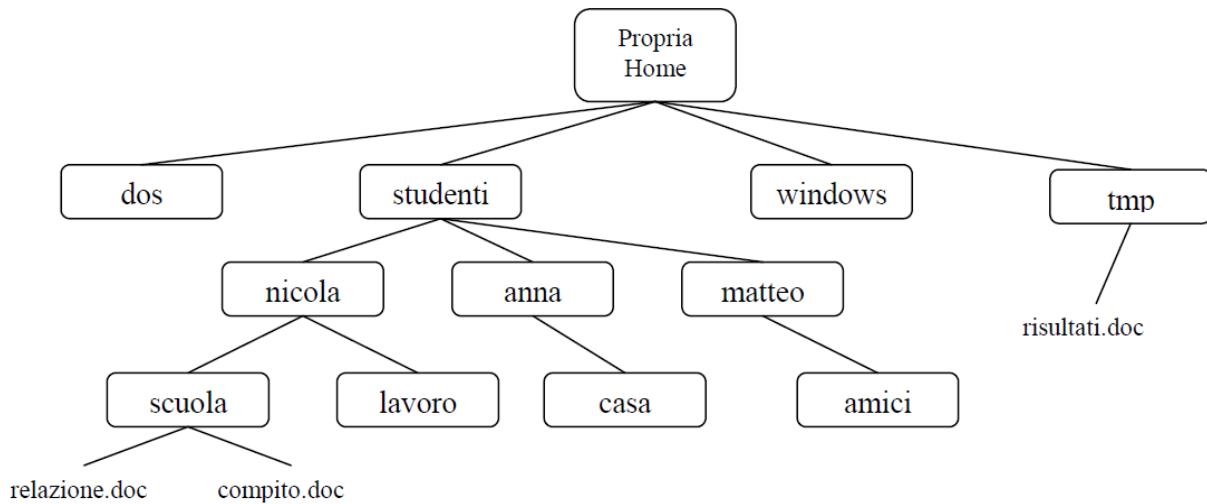
[Background & Foreground]

EPICODE - CYBERSECURITY CLASS [W5D1 Pratica2]

23 novembre 2023



Data la seguente struttura:



Ti trovi nella directory nicola (sotto studenti):

- Cambiare gli attributi della directory lavoro facendo in modo che il suo contenuto non sia leggibile ma consentendo solo a te di entrarvi
- Entra nella cartella lavoro e crea un file (nano – ricordati di salvarlo)
- Prova a visualizzare il contenuto della directory dove ti trovi dopo di che modifica gli attributi della directory '.' facendo in modo che sia nuovamente leggibile il contenuto e prova nuovamente a leggere il contenuto

```
(kali㉿kali)-[~/Desktop/esercitazione/studenti/nicola]
$ chmod 300 lavoro

(kali㉿kali)-[~/Desktop/esercitazione/studenti/nicola]
$ cd lavoro

(kali㉿kali)-[~/.../esercitazione/studenti/nicola/lavoro]
$ nano file.txt

(kali㉿kali)-[~/.../esercitazione/studenti/nicola/lavoro]
$ ls
ls: cannot open directory '.': Permission denied

(kali㉿kali)-[~/.../esercitazione/studenti/nicola/lavoro]
$ chmod 700 .

(kali㉿kali)-[~/.../esercitazione/studenti/nicola/lavoro]
$ ls
file.txt
```

- d) Spostati nella cartella scuola usando il percorso relativo (a dove ti trovi)
- e) Nella directory scuola crea una directory .mia (punto mia) e fai in modo che sia leggibile scrivibile ed eseguibile solo da te
- f) Prova a eseguire il comando ls e successivamente ls -al

```
(kali㉿kali)-[~/.../esercitazione/studenti/nicola/lavoro]
$ cd ../scuola

(kali㉿kali)-[~/.../esercitazione/studenti/nicola/scuola]
$ mkdir .mia

(kali㉿kali)-[~/.../esercitazione/studenti/nicola/scuola]
$ chmod 700 .mia

(kali㉿kali)-[~/.../esercitazione/studenti/nicola/scuola]
$ ls -al
total 12
drwxr-xr-x 3 kali kali 4096 Nov 23 09:21 .
drwxr-xr-x 4 kali kali 4096 Nov 22 13:11 ..
-rw-r--r-- 1 kali kali    0 Nov 22 13:12 compito.doc
drwx----- 2 kali kali 4096 Nov 23 09:21 .mia
-rw-r--r-- 1 kali kali    0 Nov 22 13:12 relazione.doc
```

Per proseguire occorre:

1. lanciare il comando nano &
2. eseguire il comando jobs
3. lanciare il comando firefox e successivamente sul terminale premi ^z
4. mandare il processo firefox in background (bg)
5. lanciare il comando jobs
6. mandare in foreground (fg) il programma nano
7. terminare nano
8. verificare quanto spazio si sta occupando su disco

```
(kali㉿kali)-[~/Desktop]
└─$ nano 8
[1] 303126

[1] + suspended (tty output) nano
(kali㉿kali)-[~/Desktop]
└─$ jobs
[1] + suspended (tty output) nano

(kali㉿kali)-[~/Desktop]
└─$ firefox
^Z
zsh: suspended firefox

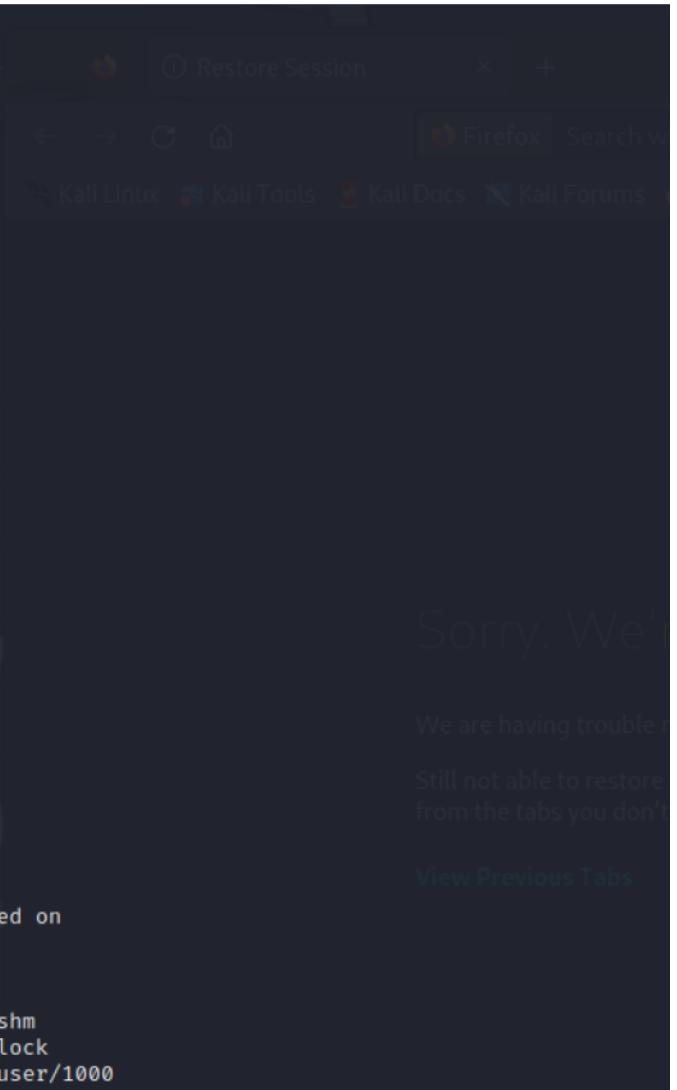
(kali㉿kali)-[~/Desktop]
└─$ jobs
[1] - suspended (tty output) nano
[2] + suspended firefox

(kali㉿kali)-[~/Desktop]
└─$ bg % 2
[2] - continued firefox
bg: job not found: 2

(kali㉿kali)-[~/Desktop]
└─$ fg %1
[1] - continued nano

(kali㉿kali)-[~/Desktop]
└─$ jobs
[2] + running firefox

(kali㉿kali)-[~/Desktop]
└─$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            945M   0    945M   0% /dev
tmpfs           197M 1016K  196M   1% /run
/dev/sda1       79G   14G   61G  19% /
tmpfs           984M   0    984M   0% /dev/shm
tmpfs           5.0M   0    5.0M   0% /run/lock
tmpfs           197M  116K  197M   1% /run/user/1000
```



PROCESSI JOB, BACKGROUND E FOREGROUND

Ogni comando che viene eseguito sulla macchina è un processo e ad esso è assegnato un numero d'ordine unico (PID). Un esempio è il comando “sleep [secondi]” che blocca l’inserimento nel terminale per il tempo specificato. Ad esempio “sleep 60” blocca il terminale per un minuto.

```
(kali㉿kali)-[~]  
$ sleep 60
```

Terminale con sleep in esecuzione


```
(kali㉿kali)-[~]  
$ sleep 60  
  
(kali㉿kali)-[~]  
$
```

Terminale dopo 60 secondi

Aperto un altro terminale e lanciando il comando “ps aux” possiamo vedere il processo in esecuzione.

```
232234 0.1 0.0 307036 10184 ? S 07:04 0:00 [kworker/1:1-0  
232236 0.0 0.5 307036 10184 ? Sl 07:04 0:00 /usr/lib/x86_64  
232249 0.0 0.0 5472 1664 pts/2 S+ 07:04 0:00 sleep 60  
232266 1.5 0.3 10180 6364 pts/3 Ss 07:04 0:00 /usr/bin/zsh  
232272 50.0 0.2 11206 4726 pts/2 R 07:04 0:00 ps aux
```

Dopo 60 secondi, il processo non è più nella lista perché è stato terminato. Se si volesse terminare il processo mentre è in esecuzione si può prendere nota del PID e lanciare il comando **sudo kill 232249**.



Abbiamo detto che il comando “sleep 60” blocca il terminale per 60 secondi o comunque per il tempo specificato. Per continuare a utilizzare lo stesso terminale nel lasso di tempo in cui il processo è in esecuzione gli scenari sono due: o si interrompe il processo (con ctrl+c o con sudo kill [PID]) oppure si esegue in background. L'esecuzione in background consente di eseguire più processi contemporaneamente rendendo quindi disponibile il terminale per altri processi.

```
(kali㉿kali)-[~]  
$ sleep 60 &  
[1] 253577  
  
(kali㉿kali)-[~]  
$ jobs  
[1] + running    sleep 60  
  
(kali㉿kali)-[~]  
$
```

Per eseguire il processo in background si chiama il comando seguito dal simbolo “&”. In questo caso, se osserviamo bene, l'esecuzione del comando “sleep 60 &” ha liberato il terminale e ha “nascosto” l'esecuzione del processo. Infatti, subito dopo siamo stati in grado di chiamare altri comandi, come il comando jobs che ci ha mostrato tutti i processi in background. Ad ogni job viene assegnato un numero.

Notare che appena viene lanciato il comando in automatico appare “[1] 253577”, vale a dire “il processo con PID 253577 (cioè “sleep 60”) è il primo tra i processi in background.

Se mettiamo un altro processo in background nel frattempo la situazione sarà questa:

```
(kali㉿kali)-[~]  
$ sleep 60 &  
[1] 261769  
  
(kali㉿kali)-[~]  
$ sleep 61 &  
[2] 261833  
  
(kali㉿kali)-[~]  
$ jobs  
[1] - running    sleep 60  
[2] + running    sleep 61
```

Per far ritornare il processo in foreground, ovvero in primo piano (come viene eseguito di default), si digita il comando “fg %[numero processo]”. Ad esempio se vogliamo riportare in primo piano il secondo job scriviamo:


```
(kali㉿kali)-[~]  
$ jobs  
[1] - running    sleep 60  
[2] + running    sleep 61  
  
(kali㉿kali)-[~]  
$ fg %2  
[2] - running    sleep 61  
█
```

Come si può notare, riportando in foreground “sleep 61” il terminale non è più utilizzabile dato che è occupato dal processo. Quando il job si conclude (in questo caso, dopo che il tempo è passato), ci veniamo avvisati in automatico in questo modo:

```
(kali㉿kali)-[~]  
$ sleep 2 &  
[1] 265434  
  
(kali㉿kali)-[~]  
$  
[1] + done sleep 2  
(kali㉿kali)-[~]  
$
```

Un processo lanciato si può “mettere in pausa” con il comando “ctrl + z”. Se non funziona “shift+ctrl+z”. Ad esempio:

```
(kali㉿kali)-[~]  
$ sleep 60 &  
[1] 268738  
  
(kali㉿kali)-[~]  
$ sleep 61 &  
[2] 268788  
  
(kali㉿kali)-[~]  
$ sleep 62  
^Z  
zsh: suspended sleep 62  
  
(kali㉿kali)-[~]  
$ jobs  
[1] running sleep 60  
[2] - running sleep 61  
[3] + suspended sleep 62
```

Abbiamo lanciato in sequenza i comandi “sleep 60” e “sleep 61” in background aggiungendo la “&” alla fine. Subito dopo abbiamo lanciato “sleep 62” in foreground, cioè normalmente. Il terminale pertanto era bloccato. Con la combinazione “shift+ctrl+z” abbiamo sospeso il processo. Lanciando jobs infatti si possono vedere i primi due processi in background e il terzo che abbiamo sospeso.

Per riprendere l'esecuzione del programma ci sono due possibilità: background o in foreground. Nel primo caso, basta lanciare il comando “bg [numero job]”.

```
(kali㉿kali)-[~]
$ bg %3
[3] - continued sleep 62

(kali㉿kali)-[~]
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=63.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=69.1 ms
^C
— 8.8.8.8 ping statistics —
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 63.132/66.113/69.094/2.981 ms

(kali㉿kali)-[~]
$ jobs
[1]  running    sleep 60
[2]  + running    sleep 61
[3]  - running    sleep 62
```

Lanciando “bg %3” il processo riprende in background. Per prova, durante l'esecuzione abbiamo usato altri comandi (ping) e richiamato jobs per verificare che il processo fosse in background nel mentre.

Nel secondo caso, basta richiamare il comando in foreground come abbiamo già visto. Come si nota nella foto sotto, richiamandolo in questa maniera il terminale si blocca e preclude il lancio di altri comandi.

```
(kali㉿kali)-[~]  
$ jobs  
[1]      running      sleep 60  
[2]  -  running      sleep 61  
[3]  +  suspended     sleep 62  
  
(kali㉿kali)-[~]  
$ fg %3  
[3]  -  continued     sleep 62  
█
```